

## **REMARKS/ARGUMENTS**

Amendments were made to the specification to correct errors noted by the Examiner. No new matter has been added by any of the amendments to the specification.

Claims 1, 2, 4-10, 12-18, and 20-29 are pending in the present application. By the present response, claims 1, 9, 17, 21, and 25 are amended. Reconsideration of the claims is respectfully requested.

### **I. Objection to the Drawings**

The Examiner objected to Figure 6, which was mentioned in the specification, but not present in the drawings. In response, the reference to Figure 6 has been deleted from the specification. This objection is overcome.

### **II. 35 U.S.C. § 103, Obviousness**

#### **II(A). Claims 1-2, 4-6, 9, 10, 12-24, 17, 18, and 20-22**

Claims 1-2, 4-6, 9, 10, 12-24, 17, 18, and 20-22 stand rejected under 35 U.S.C. §103 as being obvious over **Davis et al.** (US 5,495,577), hereinafter referred to as **Davis**, in view of **Call** (US 2002/0143521). This rejection is respectfully traversed.

The rejection states that:

**Regarding claim 1**, Davis teaches a method for "retrieving a data value from a character stream" by processing a text stream and obtaining information for each character in the data (text) stream (see col. 6, ll. 14-16). Davis teaches validity determination by determining if a character is found in a valid font type. Taking broadest reasonable interpretation of the claim as written, it is unclear as to what type of validity is actually being tested and therefore Davis' test of validity falls within the scope of the claim (see col. 6, ll. 16-23). Davis teaches on performing a validity test on each character in the stream but does not clearly recite the explicit use of a data structure to store the characters. However, in related art, Call teaches on this aspect. Call teaches the use of a data structure, an array, to store and index using integer values of character data (p. 2, para. 0016). One of ordinary skill in the art at the time of the applicant's invention would have found it obvious to utilize a data structure like an array to index character values as demonstrated by Call in combination with the character validation method taught and suggested by Davis. One of ordinary skill in the art would have been motivated to utilize a data structure like an array to promote easy organization and efficient execution of processing functions by way of easy indexing of character values (see Call, p. 2, para. 0016).

Office action mailed August 8, 2006, pp 3-4.

Claim 1 has now been amended to recite that the determining step determines the data value's validity as a character used in a given computer language. This amendment is supported by the specification at, e.g., page 6, lines 24-26 and page 7, line 29 through page 8, line 1. Claim 1 now reads:

1. A computer-implemented character validation method comprising the steps of:  
retrieving a data value from a character stream; and  
determining a validity of a character represented by said data value by locating a member of a data structure, said member having a direct correspondence to said data value, wherein said validity is determined according to a logical combination of a plurality of status values in said member of said data structure, wherein the determining step determines the data value's validity as a character within a given computer language.

The determination of "nonobviousness" is made after establishing the scope and content of prior art, the differences between the prior art and the claims at issue, and the level of ordinary skill in the pertinent art. *Graham v. John Deere*, 383 U.S. 1 (1966). In addition, all limitations of the claimed invention must be considered when determining patentability. *In re Lowry*, 32 F.3d 1579, 1582, 21 U.S.P.Q.2d 1031, 1034 (Fed Cir. 1994).

It is submitted that no *prima facie* assertion of obviousness can be made against amended claim 1 using the references relied on, because the proposed combination would not meet the invention as now claimed. Specifically, neither **Davis** nor **Call** discloses or suggests "*wherein said validity is determined according to a logical combination of a plurality of status values in said member of said data structure*" or "*wherein the determining step determines the data value's validity as a character within a given computer language*".

#### **Logical Combination of Status Values**

In the situation addressed by the present application, a given character is valid if the character is one of the valid data types present in the language. One method of determining validity is to test the character against each of the possible data types. The invention recited in claim 1 eliminates the need to perform separate determinations for each possible data type. By using the claimed data structure containing a plurality of status values, a determination can be made in a single step whether a data value is an acceptable character, no matter how many data types the language recognizes. More complicated (and time-consuming) tests can be postponed until all characters have been recognized as valid.

**Call** is cited as showing the use of a data structure, such as an array, to determine validity of a character. The cited excerpt states:

[0016] It is a further object of the invention to store both fixed and variable length data as an addressable array of integer values organized to permit more efficient execution of processing functions of the type typically performed by database management systems.

**Call**, paragraph 0016.

Although **Call** utilizes an array, this reference does not disclose “*status values*” within the array that are used to recognize valid characters. In **Call**, the array is utilized by a method “*for storing and transmitting ... data and for performing processing operations on such data*” (**Call**, paragraph 0004). Neither does **Call** or **Davis** provide any suggestion that would make a determination of validity easier. Since **Call** does not disclose or suggest a “*plurality of status values*”, this reference cannot disclose or suggest “*wherein said validity is determined according to a logical combination of a plurality of status values in said member of said data structure*”. Additionally, **Davis** also does not disclose or suggest this feature, nor does the rejection suggest otherwise. Therefore this feature is not met by either of the references relied on.

#### **Character’s Validity within a Computer Language**

The excerpt of **Davis** that is cited as checking validity of the character states:

FIG. 3 sets, forth [sic] the detailed logic for finding a font in accordance with the subject invention. Processing commences at function block 310 where the text stream is obtained and a pointer to the initial character is identified. Then, at decision block 320, each character y from the initial character to the beginning of the text is checked and a determination is made at decision block 330 to determine if the character is available in the y’s font. If the character is available, then the font is returned at terminal 370. If the character is not available, then decision block 320 is executed to determine if additional characters remain for processing. When all characters have been processed backward, then another loop is commenced at decision block 340 for each character x from z until the end of the text to determine at decision block 350 if z is available in x’s font. If a font is available, then the font is returned at terminal 370. If not, then the next character is processed at decision block 340 until all characters are processed. Then, at function block 360, a list of fonts is created and at decision block 380, the current character is checked against each font in the list to attempt to identify a mapping. If a mapping is obtained, then the font is returned via terminal 370. If no mapping is obtained, then font not found is returned via terminal 390.

**Davis**, column 6, ll. 13-35, emphasis added.

**Davis** determines whether characters are available in specific fonts. This excerpt is concerned with finding a common font in which all characters in the string can be represented. The font in which a data value is displayed is not related to the “*data value’s validity as a character within a given computer language*”. Validity as a character within a computer language has to do with the meaning, or lack thereof, assigned to a data value within the context of the computer language. Determining whether a character is available in a given font does not determine whether the character has a contextual meaning; fonts are relevant only to display characteristics. Therefore, **Davis** does not disclose or suggest “*wherein the determining step determines the data value’s validity as a character within a given computer language*”. Additionally, **Call** also does not disclose or suggest this feature, nor has the rejection suggested otherwise. Thus, this feature is not met by any of the claims relied on.

Since two of the features of the invention recited in claim 1 are not met, the rejection of claim 1 is overcome. Additionally, claims 9 and 17 are rejected for the same reasons as claim 1, so the rejection of these claims is also overcome. Additionally, claims 2, 4-6, 10, 12-16, 18 and 20-22 are dependent on one of claims 1, 9, and 17, so the rejection of these claims is likewise also overcome. Further, the dependent claims recite additional features that provide further distinctions over the reference cited.

For example, claim 5 recites that, "if each character in said character stream is valid, applying a predetermined set of syntactic rules to byte patterns comprising said character stream". Against this feature, the rejection cites the following excerpt from **Davis**, "If a font is available, then the font is returned at terminal 370. If not, then the next character is processed at decision block 340 until all characters are processed" (**Davis**, column 6, lines 27-30). **Davis** does not apply a set of syntactic rules, which Merriam-Webster's online dictionary defines as rules "of, relating to, or according to the rules of syntax or syntactics". The syntax of a language deals with how elements of the language are properly combined. This quote from **Davis**, on the other hand, does not deal with combinations of the characters that are valid, but only with asserting that a common font has, or has not, been found. Thus, the features of this claim are not met.

Therefore, the rejection of claims 1-2, 4-6, 9-10, 12-24, 17-18, and 20-22 under 35 U.S.C. §103 has been overcome.

## **II(B). Claims 7-8, 15-16 and 23-25**

Claims 7-8, 15-16 and 23-25 stand rejected under 35 U.S.C. 103(a) as being obvious over **Davis** and **Call** in view of **Zhao et al.** (US 2002/0042707 A1), hereinafter referred to as **Zhao**.

The rejection states:

**Regarding claims 7 and 8**, **Davis** teaches the use of a wide range of fonts and styles but does not explicitly disclose the use of extensible markup language (XML) syntax. However **Zhao** teaches the analysis and format determination of extensible markup language (XML) (see fig. 6, grammar packaging). At the time of the applicant's invention, it would have been obvious to one of ordinary skill in the art to modify **Davis**'s method to allow it to process XML documents as input, as taught by **Zhao**. It logically follows that the rules employed by **Davis**'s character validation would be in accordance with extensible markup language (XML) also. The motivation for doing so would have been to be able to determine whether extensible markup language (XML) packets match the extensible markup language (XML) protocol definition at an increased speed over prior methods. Therefore it would have been obvious to combine **Davis**, **Call** and **Zhao** for the benefit of increased processing speed to obtain the invention as specified in claims 7-8.

Office Action mailed August 8, 2006, pp 6-7.

The claims in this rejection are each dependent on one of claims 1, 9, and 17, discussed in the earlier rejection. Since the rejection of the independent claims has been overcome, the rejection of these dependent claims is also overcome. Specifically, neither **Davis** nor **Call** disclose or suggest the features “wherein said validity is determined according to a logical combination of a plurality of status values in said member of said data structure” or “wherein the determining step determines the data value’s validity as a character within a given computer language”, as discussed with regard to exemplary claim 1.

Additionally, **Zhao** does not disclose the features of claim 7 and also does not disclose the features of claim 8, which is dependent on claim 7. Claim 7 recites, “wherein said character stream comprises characters in accordance with a specification for an extensible markup language, and wherein said status values are set in accordance with a set of valid characters defined in said specification”. Although the rejection asserts that **Zhao** uses XML, Applicants could find no evidence of this assertion in this reference. Figure 6 of **Zhao**, cited against this claim, and is reproduced below, along with a description of the figure.

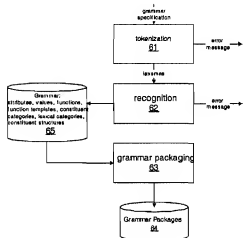


Fig. 6

[0176] FIG. 6 is a block diagram of a grammar package compiler in accordance with an embodiment of the present invention. The grammar specification (an SFG file) is input to the tokenization process 61 to separate the various lexemes in the SFG file (see for example FIG. 5). The tokenization process checks that the SFG file follows the correct format and produces error messages when the SFG file format is incorrect. The lexemes are then used in the recognition process 62 to create an internal representation of the grammar (65) comprising all attributes, values, functions, function templates, constituent categories, lexical categories and constituent structures. The recognition process will check that the SFG description is valid, for example that constituent structures only use constituent categories that are defined etc. On detection of errors an appropriate error message is generated. The grammar packaging process (63) then builds all possible grammar packages (representing phrase structure trees) that meet the descriptions and constraints described by the grammar and by the optional constraints on packages, such as width and depth of the resulting packages. The grammar packages that meet the constraints are stored in the grammar package database (64) which can be further optimally organized for fast retrieval and access by the parser process that will use the grammar packages.

**Zhao**, paragraph 0176.

There is no mention of XML in this excerpt, and indeed, the undersigned agent could find no reference to XML anywhere in this reference. Instead, the following definitions from **Zhao** are noted:

[0086] "Language usage" refers to written or spoken language and therefore includes text and speech. . . .

[0089] A "Structure Function Grammar" (SFG) is a grammar that describes both structural and relational dimensions of syntax.

**Zhao**, paragraphs 0086, 0089.

**Zhao** does not utilize XML or any other markup language; instead this reference processes the grammar of "natural" languages that are spoken or written, not computer languages. The rejection acknowledges that **Davis** does not disclose XML. **Call** mentions XML, but does not disclose the use of status values to denote whether a data value represents a valid XML character. None of the references relied on disclose *"wherein said character stream comprises characters in accordance with a specification for an extensible markup language, and wherein said status values are set in accordance with a set of valid characters defined in said specification"*. Therefore, no *prima facie* case of obviousness can be made against these claims using the references relied on and the rejection of claims 7-8, 15-16 and 23-25 is overcome.

## **II(C). Claims 26-29**

Claims 26-29 are rejected under 35 U.S.C. 103(a) as being obvious over **Davis**, **Call** and **Zhao** in view of **Jurion et al.** (US 6,631,501 B1), hereinafter referred to as **Jurion**.

The rejection states:

**Regarding claims 26-29**, the combination of **Davis**, **Call** and **Zhao** as outlined in the above rejections teaches upon the aspects of character stream parsing and performing validity tests upon the parsed characters but does not clearly teach upon the aspect wherein the parsed characters are tested to be "base" characters, "digit" characters and "extender" characters. While **Davis**, **Call** and **Zhao** do teach upon the usage of characters in general (i.e. **Davis** is testing font characters), nothing is explicitly recited to classify these characters into general groups (i.e. base, digit and extender). However, in related art, **Jurion** teaches the automatic and replacement of characters wherein characters are tested on their validity within a group or string of characters to determine whether a character within the string is appropriate, or valid. **Jurion** teaches that the characters analyzed can be of a plurality of different types of characters which would implicitly include "base" characters, "digit" characters, and "extender" characters as claimed by applicant and therefore one of ordinary skill in the art at the time of the applicant's invention would have found it obvious to test the validity of characters utilizing aspects taught by **Jurion** specifically the use of base, digit, and extender characters (col. 3, lines 8-18). One of ordinary skill in the art would have been motivated to utilize the teachings of **Jurion** in combination with the teachings of **Davis**, **Call**, and **Zhao** in order to check the

syntactical rules of character streams correctly and efficiently as provided by **Jurion** as a necessary need in the art of simple character validation (see **Jurion**, col. 2, 11. 41 -52).

Office Action dated August 8, 2006, pp. 2-7.

Claim 26 recites

26. The method of claim 25 wherein said character stream comprises a message packaged in accordance with an extensible markup language, said first status value indicates whether said data value is a valid base character, said second status value indicates whether said data value is a valid digit character, and a third status value indicates whether said data value is a valid extender character.

The cited excerpt from **Jurion** states:

Generally described, when a user types simple characters in the formation of complex characters in selected languages, such as Thai, Hindi and Vietnamese, a determination is made as to whether each newly typed simple character may be added to the sequence of characters already typed by the user. If adding the new character to the sequence violates the rules of the selected language, an attempt is made to replace an existing character in the previously typed sequence with the new character or to insert the new character at a position within the previously typed sequence in a manner that does not violate the rules.

**Jurion**, column 3, lines 8-18.

Although **Jurion** is making a determination about a character that is entered, this excerpt has nothing to do with determining whether the character is one of several types of characters in a computer language. **Jurion** uses natural languages, not computer languages. Further, the types of characters recognized in **Jurion** are not the same character types as shown in XML. Instead of the claimed base characters, digit characters, and extender characters, the Asian languages that are utilized in **Jurion** have “vowels, consonants, diacritics, tone marks, and accents” (**Jurion**, column 1, lines 41-42). This set of claims does not broadly recite character types used in any language, but recites specific character types used in XML. Neither do any of the other references relied on show the use of the claimed types of characters, as the rejection acknowledges. Therefore, since none of the references discloses or suggests the features of these claims, the rejection of claims 26-29 has been overcome.

### **III. Conclusion**

It is respectfully urged that the subject application is patentable over the art relied on and is now in condition for allowance.

The Examiner is invited to call the undersigned at the below-listed telephone number if in the opinion of the Examiner such a telephone conference would expedite or aid the prosecution and examination of this application.

DATE: November 2, 2006

Respectfully submitted,

/Betty Formby/

Betty Formby  
Reg. No. 36,536  
Yee & Associates, P.C.  
P.O. Box 802333  
Dallas, TX 75380  
(972) 385-8777  
Agent for Applicants